Package ‘Dforest’

November 28, 2017

Type Package

Title Decision Forest

Version 0.4.2

Date 2017-11-28

Author Leihong Wu <leihong.wu@fda.hhs.gov>,
       Weida Tong (Weida.tong@fda.hhs.gov)

Maintainer Leihong Wu <leihong.wu@fda.hhs.gov>

Depends R (>= 3.0)

Imports rpart, ggplot2, methods, stats

Description Provides R-implementation of Decision forest algorithm, which combines the predictions of multiple independent decision tree models for a consensus decision. In particular, Decision Forest is a novel pattern-recognition method which can be used to analyze: (1) DNA microarray data; (2) Surface-Enhanced Laser Desorption/Ionization Time-of-Flight Mass Spectrometry (SELDI-TOF-MS) data; and (3) Structure-Activity Relation (SAR) data.

In this package, three fundamental functions are provided, as (1) DF_train, (2) DF_pred, and (3) DF_CV. run Dforest() to see more instructions.


License GPL-2

LazyLoad yes

LazyData yes

Encoding UTF-8

NeedsCompilation no

RoxygenNote 6.0.1

Repository CRAN

Date/Publication 2017-11-28 22:03:57 UTC
R topics documented:

- cal_MCC ............................................. 2
- Con_DT ............................................... 3
- data_dili ........................................... 3
- Dforest ............................................. 4
- DF_acc ............................................... 4
- DF_calp ............................................ 5
- DF_ConfPlot ........................................ 5
- DF_ConfPlot_accu .................................. 6
- DF_CV ................................................ 6
- DF_CVsummary ...................................... 8
- DF_dataFs .......................................... 8
- DF_dataPre ......................................... 9
- DF_easy ............................................. 9
- DF_perf ............................................. 10
- DF_pred ............................................ 11
- DF_train ........................................... 12
- DF_Trainsummary ................................. 13
- multiplot .......................................... 13
- Pred_DT ........................................... 14

Index 15

| cal_MCC                  | Performance evaluation from other modeling algorithm Result |

Description

Performance evaluation from other modeling algorithm Result

Usage

cal_MCC(pred, label)

Arguments

- pred       Predictions
- label      Known-endpoint

Value

- result$ACC: Predicting Accuracy
- result$MIS: MisClassification Counts
- result$MCC: Matthew’s Correlation Coefficients
- result$bACC: balanced Accuracy
**Con_DT**

*Construct Decision Tree model with pruning*

**Description**

Construct Decision Tree model with pruning

**Usage**

```
Con_DT(X, Y, min_split = 10, cp = 0.01)
```

**Arguments**

- **X**: dataset
- **Y**: data_Labels
- **min_split**: minimum number of node in each leaf
- **cp**: pre-defined Complexity Parameter (CP) rpart program

**Value**

Decision Tree Model with pruning Implemented by rpart

**See Also**

rpart

---

**data_dili**

*QSAR dataset with DILI endpoint for demo*

**Description**

This data set gives the DILI endpoint of various compounds (Most or No DILI-concern) with QSAR descriptors generated by MOLD2

**Usage**

```
rivers
```

**Format**

A List containing two vectors: X contains 958 observations and 777 variables. Y contains DILI endpoints of 958 observations

**Source**

In-house data
References

Minjun Chen (2011) *FDA-approved drug labeling for the study of drug-induced liver injury.* Drug discovery today

Dforest

Demo script to lean Decision Forest package Demo data are located in data/ folder

Description

Demo script to lean Decision Forest package Demo data are located in data/ folder

Usage

Dforest()

Author(s)

Leihong.Wu

Examples

Dforest()

DF_acc

Performance evaluation from Decision Tree Predictions

Description

Performance evaluation from Decision Tree Predictions

Usage

DF_acc(pred, label)

Arguments

pred Predictions
label Known-endpoint

Value

result$ACC: Predicting Accuracy
result$MIS: MisClassification Counts
result$MCC: Matthew’s Correlation Coefficients
result$bACC: balanced Accuracy
**DF_calp**

*T-test for feature selection*

**Description**

T-test for feature selection

**Usage**

```R
DF_calp(x, y)
```

**Arguments**

- `x`: X variable matrix
- `y`: Y label

---

**DF_ConfPlot**

*Decision Forest algorithm: confidence level accumulated plot*

**Description**

Draw accuracy curve according to the confidence level of predictions

**Usage**

```R
DF_ConfPlot(Pred_result, Label, bin = 20, plot = T, smooth = F)
```

**Arguments**

- `Pred_result`: Predictions
- `Label`: known label for Test Dataset
- `bin`: How many bins occurred in Conf Plot (Default is 20)
- `plot`: Draw Plot if True, otherwise output the datamatrix
- `smooth`: if TRUE, Fit the performance curve with smooth function (by ggplot2)

**Value**

- `ACC_Conf`: return data Matrix ("ConfidenceLevel", "Accuracy", "Matched Samples") for confidence plot (no plot)
- `ConfPlot`: Draw Confidence Plot if True, need install ggplot2
**DF_ConfPlot_accu**

*Decision Forest algorithm: confidence level accumulated plot (accumulated version)*

**Description**

Draw accuracy curve according to the confidence level of predictions

**Usage**

\[
\text{DF_ConfPlot_accu(Pred_result, Label, bin = 20, plot = T, smooth = F)}
\]

**Arguments**

- **Pred_result**: Predictions
- **Label**: known label for Test Dataset
- **bin**: How many bins occurred in Conf Plot (Default is 20)
- **plot**: Draw Plot if True, otherwise output the datamatrix
- **smooth**: if TRUE, Fit the performance curve with smooth function (by ggplot2)

**Value**

- **ACC_Conf**: return data Matrix ("ConfidenceLevel", "Accuracy", "Matched Samples") for confidence plot (no plot)
- **ConfPlot**: Draw Confidence Plot if True, need install ggplot2

---

**DF_CV**

*Decision Forest algorithm: Model training with Cross-validation*

**Description**

Decision Forest algorithm: Model training with Cross-validation Default is 5-fold cross-validation

**Usage**

\[
\text{DF_CV(X, Y, stop_step = 10, CV_fold = 5, Max_tree = 20, min_split = 10, cp = 0.1, Filter = F, p_val = 0.05, Method = "bACC", Quiet = T, Grace_val = 0.05, imp_accu_val = 0.01, imp_accu_criteria = F)}
\]
**Arguments**

- **X**  
  Training Dataset

- **Y**  
  Training data endpoint

- **stop_step**  
  How many extra step would be processed when performance not improved, 1 means one extra step

- **CV_fold**  
  Fold of cross-validation (Default = 5)

- **Max_tree**  
  Maximum tree number in Forest

- **min_split**  
  minimum leaves in tree nodes

- **cp**  
  parameters to pruning decision tree, default is 0.1

- **Filter**  
  doing feature selection before training

- **p_val**  
  P-value threshold measured by t-test used in feature selection, default is 0.05

- **Method**  
  Which is used for evaluating training process. MIS: Misclassification rate; ACC: accuracy

- **Quiet**  
  if TRUE (default), don’t show any message during the process

- **Grace_val**  
  Grace Value in evaluation: the next model should have a performance (Accuracy, bACC, MCC) not bad than previous model with threshold

- **imp_accu_val**  
  improvement in evaluation: adding new tree should improve the overall model performance (Accuracy, bACC, MCC) by threshold

- **imp_accu_criteria**  
  if TRUE, model must have improvement in accumulated accuracy

**Value**

- `.performance`: Overall training accuracy (Cross-validation)
- `.pred`: Detailed training prediction (Cross-validation)
- `.detail`: Detailed usage of Decision tree Features/Models and their performances in all CVs
- `.Method`: pass evaluating Methods used in training
- `.cp`: pass cp value used in training decision trees

**Examples**

```r
##data(iris)
X = iris[,1:4]
Y = iris[,5]
names(Y)=rownames(X)

random_seq=sample(nrow(X))
split_rate=3
split_sample = suppressWarnings(split(random_seq,1:split_rate))
Train_X = X[-random_seq[split_sample[[1]]]],
Train_Y = Y[-random_seq[split_sample[[1]]]]

CV_result = DF_CV(Train_X, Train_Y)
```
DF_CVsummary  
output summary for Dforest Cross-validation results

Description
Draw plot for Dforest Cross-validation results

Usage
DF_CVsummary(CV_result, plot = T)

Arguments
CV_result  Training Dataset
plot  if TRUE (default), draw plot

DF_dataFs  
Decision Forest algorithm: Feature Selection in pre-processing

Description
Decision Forest algorithm: feature selection for two-class predictions, kept statistical significant features pass the t-test

Usage
DF_dataFs(X, Y, p_val = 0.05)

Arguments
X  Training Dataset
Y  Training Labels
p_val  Correlation Coefficient threshold to filter out high correlated features; default is 0.95

Value
Keep_feat: qualified features in data matrix after filtering

Examples
##data(iris)
X = iris[iris[,5]=="setosa",1:4]
Y = iris[iris[,5]=="setosa",5]
used_feat = DF_dataFs(X, Y)
DF_dataPre

**Decision Forest algorithm: Data pre-processing**

**Description**
Decision Forest algorithm: Data pre-processing, remove All-Zero columns/features and high correlated features

**Usage**
DF_dataPre(\(X, \text{thres} = 0.95\))

**Arguments**
- \(X\): Training Dataset
- \(\text{thres}\): Correlation Coefficient threshold to filter out high correlated features; default is 0.95

**Value**
- \(\text{Keep\_feat}\): qualified features in data matrix after filtering

**Examples**
```r
#data(iris)
X = iris[,1:4]
Keep\_feat = DF_dataPre(X)
```

DF_easy

**Simple pre-defined pipeline for Decision forest**

**Description**
This is a script of decision forest for easy use

**Usage**
DF_easy(\(Train\_X, \text{Train\_Y}, \text{Test\_X}, \text{Test\_Y}, \text{mode} = \text{"default"}\))

**Arguments**
- \(Train\_X\): Training Dataset
- \(Train\_Y\): Training data endpoint
- \(Test\_X\): Testing Dataset
- \(Test\_Y\): Testing data endpoint
- \(\text{mode}\): pre-defined modeling
Value
data_matrix training and testing result

Examples

```r
# data(demo_simple)
X = iris[,1:4]
Y = iris[,5]
names(Y)=rownames(X)

random_seq=sample(nrow(X))
split_rate=3
split_sample = suppressWarnings(split(random_seq,1:split_rate))
Train_X = X[-random_seq[split_sample[[1]]],]
Train_Y = Y[-random_seq[split_sample[[1]]]]
Test_X = X[random_seq[split_sample[[1]]],]
Test_Y = Y[random_seq[split_sample[[1]]]]

Result = DF_easy(Train_X, Train_Y, Test_X, Test_Y)
```

DF_perf

performance evaluation between two factors

Description

performance evaluation between two factors

Usage

DF_perf(pred, label)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pred</td>
<td>Predictions</td>
</tr>
<tr>
<td>label</td>
<td>Known-endpoint</td>
</tr>
</tbody>
</table>

Value

- `result$ACC`: Predicting Accuracy
- `result$MIS`: MisClassification Counts
- `result$MCC`: Matthew’s Correlation Coefficients
- `result$bACC`: balanced Accuracy
**Decision Forest algorithm: Model prediction**

**Description**

Decision Forest algorithm: Model prediction with constructed DF models. DT_models is a list of Decision Tree models (rpart.objects) generated by DF_train(). DT_train_CV() is only designed for Cross-validation and won’t generate models.

**Usage**

```r
DF_pred(DT_models, X, Y = NULL)
```

**Arguments**

- `DT_models`: Constructed DF models
- `X`: Test Dataset
- `Y`: Test data endpoint

**Value**

- `.s_accuracy`: Overall test accuracy
- `.s_predictions`: Detailed test prediction

**Examples**

```r
# data(demo_simple)
X = data_dili$X
Y = data_dili$Y
names(Y)=rownames(X)

random_seq=sample(nrow(X))
split_rate=3
split_sample = suppressWarnings(split(random_seq,1:split_rate))
Train_X = X[-random_seq[split_sample[1]]],
Train_Y = Y[-random_seq[split_sample[1]]]
Test_X = X[random_seq[split_sample[1]]],
Test_Y = Y[random_seq[split_sample[1]]]

used_model = DF_train(Train_X, Train_Y)
Pred_result = DF_pred(used_model,Test_X,Test_Y)
```
Description

Decision Forest algorithm: Model training

Usage

\[
\text{DF}\_\text{train}(X, Y, \text{stop}\_\text{step} = 5, \text{Max}\_\text{tree} = 20, \text{min}\_\text{split} = 10, \text{cp} = 0.1, \text{Filter} = \text{F}, \text{p}\_\text{val} = 0.05, \text{Method} = "bACC", \text{Quiet} = \text{T}, \text{Grace}\_\text{val} = 0.05, \text{imp}\_\text{accu}\_\text{val} = 0.01, \text{imp}\_\text{accu}\_\text{criteria} = \text{F})
\]

Arguments

- \text{X}: Training Dataset
- \text{Y}: Training data endpoint
- \text{stop}\_\text{step}: How many extra step would be processed when performance not improved, 1 means one extra step
- \text{Max}\_\text{tree}: Maximum tree number in Forest
- \text{min}\_\text{split}: minimum leaves in tree nodes
- \text{cp}: parameters to pruning decision tree, default is 0.1
- \text{Filter}: doing feature selection before training
- \text{p}\_\text{val}: P-value threshold measured by t-test used in feature selection, default is 0.05
- \text{Method}: Which is used for evaluating training process. MIS: Misclassification rate; ACC: accuracy
- \text{Quiet}: if TRUE (default), don’t show any message during the process
- \text{Grace}\_\text{val}: Grace Value in evaluation: the next model should have a performance (Accuracy, \text{bACC}, \text{MCC}) not bad than previous model with threshold
- \text{imp}\_\text{accu}\_\text{val}: improvement in evaluation: adding new tree should improve the overall model performance (Accuracy, \text{bACC}, \text{MCC}) by threshold
- \text{imp}\_\text{accu}\_\text{criteria}: if TRUE, model must have improvement in accumulated accuracy

Value

- \text{accuracy}: Overall training accuracy
- \text{pred}: Detailed training prediction (fitting)
- \text{detail}: Detailed usage of Decision tree Features/Models and their performances
- \text{models}: Constructed (list of) Decision tree models
- \text{Method}: pass evaluating Methods used in training
- \text{cp}: pass cp value used in training decision trees
Examples

```r
# data(iris)
X = iris[,1:4]
Y = iris[,5]
names(Y) = rownames(X)
used_model = DF_train(X, factor(Y))
```

---

**DF_Trainsummary**

output summary for Dforest test results

**Description**

Draw plot for Dforest test results

**Usage**

```r
DF_Trainsummary(used_model, plot = T)
```

**Arguments**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>used_model</td>
<td>Training result</td>
</tr>
<tr>
<td>plot</td>
<td>if TRUE (default), draw plot</td>
</tr>
</tbody>
</table>

---

**multiplot**

multiplot

**Description**

Multiple plot function

If the layout is something like `matrix(c(1,2,3,3), nrow=2, byrow=TRUE)`, then plot 1 will go in the upper left, 2 will go in the upper right, and 3 will go all the way across the bottom.

**Usage**

```r
multiplot(..., plotlist = NULL, cols = 1, layout = NULL)
```

**Arguments**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>ggplot objects</td>
</tr>
<tr>
<td>plotlist</td>
<td>a list of ggplot objects</td>
</tr>
<tr>
<td>cols</td>
<td>Number of columns in layout</td>
</tr>
<tr>
<td>layout</td>
<td>A matrix specifying the layout. If present, 'cols' is ignored.</td>
</tr>
</tbody>
</table>
**Description**
Doing Prediction with Decision Tree model

**Usage**
```r
Pred_DT(model, X)
```

**Arguments**
- `model` Decision Tree Model
- `X` dataset

**Value**
Decision Tree Predictions Different endpoints presented in multiple columns

**Source**
rpart

**See Also**
rpart
Index

*Topic datasets
  data_dili, 3

  cal_MCC, 2
  Con_DT, 3

  data_dili, 3
  DF_acc, 4
  DF_calp, 5
  DF_ConfPlot, 5
  DF_ConfPlot_accu, 6
  DF.CV, 6
  DF.CVsummary, 8
  DF.dataFs, 8
  DF.dataPre, 9
  DF.easy, 9
  DF.perf, 10
  DF.pred, 11
  DF.train, 12
  DF.TrainSummary, 13
  Dforest, 4

  multiplot, 13

  Pred_DT, 14